実世界プログラミングのための分散人力処理環境

馬場　匠見† 　橋本　翔† 　増井　俊之††

† 慶應義塾大学政策・メディア研究科　〒 252-0882　神奈川県藤沢市遠藤 5322
†† 慶應義塾大学環境情報学部　〒 252-0882　神奈川県藤沢市遠藤 5322
E-mail: †{bb,shokai}@sfc.keio.ac.jp, ††masui@pitecan.com

あらまし　コンピュータの動作の手順書としてプログラムが、人間の行動の手順書としてマニュアルやレシピといったものが存在するが、両者を同一のフォーマットで記述することは実現されていない。本論文では、プログラム上で人の行動を記述するためのプログラミング環境 BabaScript を提案する。また、提案するプログラミング環境の応用例について述べ、考察を行う。BabaScript は、人への命令構文と命令を受け取り、値を返すことのできるクライアントアプリケーションを組み合わせることで実現する。BabaScript を用いることで、人とコンピュータの処理をプログラムという同一のフォーマットで記述し、実行することができるようになる。

キーワード　ヒューマンコンピュテーション, プログラミング環境, 実世界プログラミング

1. はじめに

コンピュータに実行させたい処理を記述するための手順書として、プログラムが存在する。プログラムを実行することによって、人はコンピュータに様々な計算を行わせ、目的の計算結果を得ている。一方、人間に実行させたい処理を記述するための手順書としては、レシピやマニュアルといったものが存在する。レシピやマニュアルに従って行動することによって、人は適切・効率的に動き、目的を達成することができる。

レシピやマニュアルは、プログラムと大きく類似している。双方ともに実行する処理について記述されたものであり、プログラムはコンピュータによって解釈・実行され、レシピやマニュアルは人によって解釈・実行される。例えば、料理のレシピをプログラムのように記述するならば、以下のようになる。

```
1 if 鍋の水が沸騰する == true
2     パスタを鍋に投入する
```

また、小売店の店員マニュアルであれば、以下のような記述方法となる。

```
1 if レジに人が並んでいる == true
2     2番レジを開ける
```

プログラムとレシピ・マニュアルは大きく類似しているが、別の存在として扱われている。近年の研究で人を計算資源として利用するといったことも提案されているが、レシピやマニュアルに記述されているような、実世界における行動を記述するためのものは存在しない。

本論文では、人の行動をプログラム内に記述可能なプログラミング環境 BabaScript を提案する。BabaScript 環境では、プログラムというフォーマット上で人とコンピュータの双方への命令記述を実現可能である。人間に実行させたい処理をプログラム内で記述しプログラムとして実行することで、実際に人に命令が配信され、通常のプログラムで関数を実行したら返り値を得られることと同様に返値を得ることができる。特殊なプログラミング言語を使わず、通常のプログラムの記述方法とほぼ同じ記述方法で、人をプログラミング要素として利用可能である。また、高い拡張性を持つため、様々なプログラムに組み込むことができる。

2. BabaScript

BabaScript は、人への命令構文をプログラミング言語に付加するライブラリと、プログラムからの命令を受け取り、処理結果をプログラムに返すクライアントライブラリを組み合わせることで実現するプログラミング環境だ。BabaScript 環境下では、プログラム内で人への命令が記述可能となる。人への命令構文ライブラリとクライアントライブラリは双方共に、簡単に拡張・組み込みが可能となっており、既存のアプリケーションでもすぐに BabaScript 環境を導入することができる。

2.1 人への命令構文ライブラリ

人への命令構文を含んだオブジェクト (以下、人オブジェクト) を宣言可能にするライブラリを実装した。人オブジェクトを宣言し、そのオブジェクトのメソッドを実行することで、人への命令が配信される。オブジェクトに定義されていないメソッドは全て人への命令として解釈され、メソッドと引数を元に人への命令内容が生成される。また、命令に対して値が返ってくると、メソッド実行時に登録したコールバック関数が実行される。

2.1.1 人オブジェクトの宣言と基本命令

例えば、以下のようなプログラムで人への命令が可能となる。

```
1 baba = new Baba.Script("baba");
2 baba.書類整理をする ({num: 5}, function(result){
3     # 値が返ってきたあとの処理を記述する
4 });
```

プログラム内で人オブジェクトを宣言するときには、第一引数にグループ ID を指定する。指定した id と同一のグループ ID を監視しているクライアントに対し、命令の配信が行われる。複数のクライアントアプリケーションが同一のグループ ID を監視している場合、特別なオプションがない場合は、命令は各クライアントアプリケーションに分散して配信される。上記

のプログラムの場合、メソッド名である ”書類整理をする” と第一引数である ”num: 5” を元にして、クライアントライブラリへの命令が生成され、クライアントへと通知される。

人への命令構文の第二引数では、クライアントから値が返ってきた時に実行するコールバック関数を指定する。このコールバック関数には引数が与えられ、引数の中には返り値そのものと、値を返したクライアントに対応した人オブジェクトが入る。値を返したクライアントに命令を送りたいときなどは、この返り値に含まれる人オブジェクトに対して、人への命令構文を実行することで再び同じクライアントに対して命令を配信することができる。

2.1.2 オプション情報の付加

人への命令メソッドの第一引数にオブジェクトを与えることによってクライアントアプリケーション側にオプションとして情報を送ることができる。オプションの例としては返り値の型指定がある。プログラムへの返り値として様々な型が考えられるが、全ての型を考慮してプログラムを記述することは難しい。返り値の型を指定することによって、プログラム側で求めている値を人に入力させることが可能だ。数値の返り値を求める場合であれば、以下のようなプログラムを書き、クライアントアプリケーション側で数値だけを入力させるインタフェースを表示させるといったことが考えられる。

```
1 baba.部屋の中には人が何人いますか ({format: "int"},
        function(result){ ... });
```

数値の他にも、文字列であったりリストの中から選択する、といったフォーマットが考えられる。

特別なオプションとして broadcast が存在する。

```
1 baba.大学内にいますか ({broadcast: 5}, function(result){
2   # ...
3 });
```

上記のように、オプションに broadcast: num を指定することによって、同じ id を持つ全クライアントに対して同様の命令を送信し、 num で指定した数だけ値が返ってきたらコールバック関数を実行するといったことが可能となる。

2.2 クライアントライブラリ

命令を受け取り、値をプログラムに返すための一連の機能をクライアント側のライブラリとして実装した。クライアント側では、BabaScript 通信用のクライアントオブジェクトを宣言し、このオブジェクトを通してプログラムと通信をする。このクライアントオブジェクトからのメッセージを受け取る関数を実装することによって、プログラマ側で自由にクライアントアプリケーションを実装できる。

以下のようなプログラムで、人への命令構文を用いたプログラムからのメッセージを受け取ることができる。

```
1 client = Baba.createClient("baba");
2 client.on("get_task", function(order){
3   # プログラムからメッセージを受け取った時の挙動を記述する
4 });
```

クライアントオブジェクトを宣言するとき、第一引数に監視対象とする ID を指定する。宣言時に指定した ID に対して、人への命令構文から命令が配信された場合、クライアントオブジェクトは命令を受け取る事ができる。一つの ID を複数のクライアントオブジェクトが監視していた場合、命令は各クライアントに分散して配信される。

client オブジェクトの on メソッドの第二引数にコールバック関数を指定することで、プログラムからの命令内容を引数にしてコールバック関数が実行される。命令を受け取った後の挙動を、プログラマが自由に定義できるようにした。これによって、プログラマはより自由度の高いプログラミングが可能となる。

メッセージを受け取る関数内において、ユーザに命令を伝え、処理結果を入力するようなインタフェースを生成してワーカーに提示する必要がある。命令内容をユーザに提示し、その命令に対する返り値を入力させる入力フォームを生成し、入力がされたら入力値をプログラムに返す、といったことが可能だ。また、クライアントオブジェクトが持つメソッド returnValue を使うことで、プログラムに処理結果を返すことができる。例えば、以下のようなプログラムが考えられる。

```
1 client.on("get_task", function(result){
2   order = new Order(result.key)
3   input = new Input(result.format)
4   input.on("submit", function(value){
5     client.returnValue(value)
6   });
7 });
```

2.3 システム利用時の流れ

BabaScript 環境では、以下のような流れで人への命令を配信され、返り値を得ることができる。

（1） プログラムで人への命令構文を実行する

（2） ネットワークを介して、適切なクライアントアプリケーションに命令が配信される

（3） クライアントアプリケーション は、命令をユーザに通知する

（4） 人が命令を処理する

（5） 人は処理結果をクライアントアプリケーションに入力する

（6） 結果をネットワークを介して、プログラムに返す

（7） プログラムは、返ってきた値を元に指定された続きの処理を実行する

2.4 実　　装

人への命令構文ライブラリとクライアントライブラリは javascript で実装した。人への命令構文ライブラリは node.js 上で動作し、クライアント用ライブラリは node.js と Web ブラウザ上で動作する。加えて、スマートフォンでの利用を想定し、Web ブラウザ上で動作する Web アプリケーションを実装した。Web アプリケーションでは、クライアントライブラリを読み込むことで、命令構文によって配信される命令を受信し、命令内容によってインタフェースを変化させ、人に対して命令の実行結果の入力を促す。システムは図 1 の通りに構成される。

2.4.1 人への命令構文

人オブジェクトにおいて定義されていないメソッドを全て人への命令として解釈し、そのメソッドに与えられる引数と共にクライアントに送信する仕組みを実装した。定義されていない

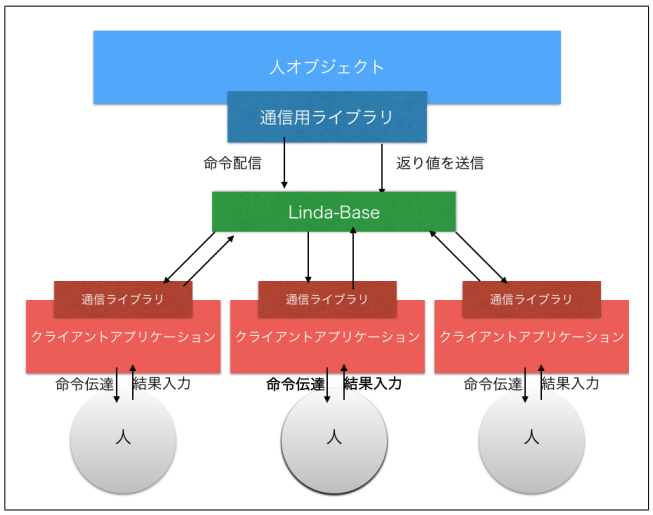

図 1　システム図

メソッドが実行された場合、代わりに他の特定のメソッドを実行する仕組みを利用してこの機能を実現した。そのため、全てのオブジェクトが持つようなメソッド名に関しては、本環境においては人への命令として機能させることはできない。

2.4.2　命令プロトコル

人への命令構文として判断されたメソッドを元に命令プロトコルを生成し、クライアントライブラリに配信している。命令プロトコルは、以下の要素から成り立つ

（1） 配信グループ ID
（2） 命令のタイプ
（3） 命令内容
（4） コールバック ID
（5） オプション

配信グループ ID は、命令を配信したいクライアント群の id を指定する。この項目で指定した id と同じ id を命令のタイプには、broadcast, unicast, eval の 3 種類が存在する。命令内容は、実行された人への命令構文のメソッド名の部分になる。コールバック ID は、ランダムに生成される文字列だ。このコールバック ID を元に、命令と返り値の関連付けを行う。オプションは、上記の基本情報以外の情報を格納するために存在する。例えば、返り値のフォーマット情報などは、オプションに含まれる。

これらを JSON 形式に変換し、タスクとしてクライアントに配信する。以下のような JSON の形式へと変換される。

```
1 task = {
2   id: "配信するクライアントの ID",
3   type: "eval",
4   key: "命令の内容",
5   cid: "ランダムの文字列",
6   options: {},
7 }
```

2.4.3　タスクの分散配信

命令を複数人に分散するために、Linda-Base [12] を利用している。Linda-Base は、ネットワークを介したノードに対して、分散的にタスクの配信が可能な Web サービスだ。

人への命令構文ライブラリも、クライアントライブラリも双方ともにこの Linda-Base にノードとして接続する。人への命令構文はこの Linda-Base にタプルと呼ばれるデータレコードを書き込む。クライアントライブラリは自分の ID と合致したタプルのみを取得し、ユーザに通知する。

ユーザが命令を受け取り、処理した結果も同様にタプルとして書き込まれる。そのタプルを人への命令構文ライブラリが取得し、クライアントからの返り値として扱う。この際、命令に対する返り値か否かは、コールバック ID が同一かどうかで判断する。

複数人への命令配信時には、各ノードはキューのような形式でリスト構造に格納され、命令配信を待つことになる。命令が配信されれば、キューの先頭のノードが命令を受信し、キューから抜け出す。命令に対して値を返すと、ノードはキューの末尾に入り、次の命令が配信されるまで待機する。

2.5　特　　徴

2.5.1　オブジェクトとして人を表現可能

普通にプログラムを書いている中で、オブジェクトを扱っているのと同じような手法で人オブジェクトを利用可能である。人オブジェクトのメソッドを実行すれば、普通のオブジェクトと同じように値が返ってくるため、普通のプログラム記述の手法で人の行動をプログラムすることができる。

2.5.2　簡単に人力処理を組み込む

人への命令構文付加ライブラリとクライアントライブラリの双方共に、簡単に既存プログラムに組み込むことが可能である。日常的に使っているプログラムに人の要素を簡単に組み込むことができ、また、様々なアプリケーションでもすぐに人力処理のワーカーになることができる。

2.5.3　外部イベントをトリガーにした人力処理

コンピュータと人の処理の双方をプログラム上で記述可能にしたことで、様々なきっかけとして、人に対して命令を配信できるようになる。家のセンサーデータの値や、Web ページの更新、日時や時間といった、今までプログラムを自動実行するために設定していたあらゆるイベントを元にして人に対して命令を配信し、人を動かすきっかけとすることができる。

2.6　プログラム例

例えば、パスタ料理を作るプログラムならば、以下のプログラム群のような記述方法が考えられる。

2.6.1　人への命令を記述したプログラム

人への命令を記述するプログラムは、以下のようなものが考えられる。

```
1
2   baba = new Baba.Script("takumibaba");
3   list = ["ペペロンチーノ", "カルボナーラ", "トマトクリーム
        ", "ボロネーゼ"];
4   baba.どれを作りますか ({format: "list", list: list},
        function(data){
5     if(data.value === "ペペロンチーノ"){
6       baba.パスタ鍋に水を入れ沸騰させる ();
7       baba.on("boil", function(){
8         baba.パスタ鍋にパスタを投入する ();
9         setTimeout(function(){
10          baba.湯切りする ()
11          baba.具材を炒めたらフライパンにパスタを投入す
                る ()
12          baba.適度に混ぜる (function(){
```

```
            baba.更に盛りつける ();
            done();
          });
        },1000*60*10);
        baba.にんにくを用意する ();
        baba.鷹の爪を用意する ();
        baba.炒める ();
      });
    }
  });
  sensor = Sensor.create("鍋")
  settInterval(function(){
    if (sensor.getState === BOIL){
      baba.emit("boil");
      clearInterval(arguments.callee);
    }
  }, 1000)
```

上記のプログラムは、「料理をする」という仕事をプログラムとして記述したものだ。「何を食べたいか」といった、人の主観的な情報が必要な部分や、「炒める」、「盛り付ける」などの人の動作が必要な部分は人への命令構文を利用している。また、温度センサーとの連携や時間計測が必要な部分においては、コンピュータに実行させるといったことが考えられる。

2.6.2 クライアント側プログラム

人への命令を受け取り、ユーザに提示するプログラムは以下のようなものが考えられる。

```
<h1 id="title"></h1>
<div id="return-value-view">
</div>
<script type="text/javascript">
  client = Baba.createClient("takumibaba").on("get_task
      ",function(data){
    title = $("title")
    title.html(data.key)
    $("#return-value-view").empty()
    if(data.format === "list"){
      select = $("select").addClass("value-list")
      for(int i=0;i<data.list.length;i++){
        option = $("<option>"+ list[i] +"</option>")
        select.append option
      }
      button = $("button").click(function(){
        client.returnValue($("value-list").val())
      });
      $("#return-value-view").append(select)
    }else{
      tButton = $("button").addClass("true-button").
          click(function(){
        client.returnValue(true)
      });
      fButton = $("button").addClass("false-button").
          click(function(){
        client.returnValue(false)
      });
      $("#return-value-view").append(tButton);
      $("#return-value-view").append(fButton);
    }
  });
</script>
```

上記のプログラムでは、料理プログラムから命令を受け取り、命令内容に応じて異なる入力インタフェースを提示して入力を促すといったことを記述している。

2.6.3 クライアント側インタフェース

クライアント側のインタフェースとしては、図 2 や 図 3 といったインタフェースが考えられる。

命令を受け取っていない場合は、図 2 のような待機画面を表示させておき、行動すべきことがないということをユーザに提示する。命令を受け取ると、待機画面から 図 3 のような画面を表示し、ユーザに行動と返り値の入力を求める。

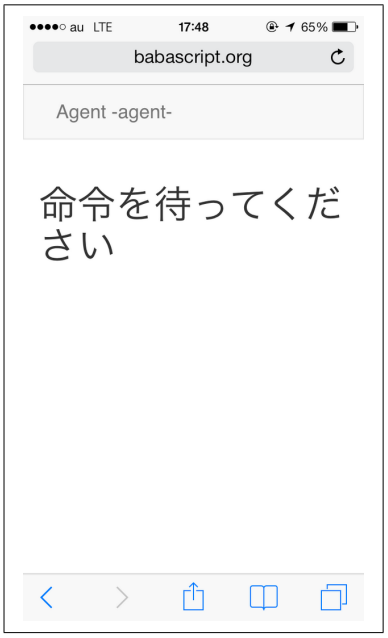

図 2　待機時の画面

図 3　リスト表示時の画面

3. 応　用　例

3.1 人の仕事や役割をプログラム化・実行可能にする

人の行動をプログラムとして記述可能になることで、人の仕事や役割がプログラム化・実行可能にするといった応用が考えられる。仕事や役割はマニュアルやドキュメントという形で言語化されていることが多い。言語化されているということは、BabaScript 環境を用いることでプログラムとして記述可能である。

プログラム化し、実行可能になれば、人はプログラムからの命令に従うだけで、記述されている内容を実行可能になる。ただ命令に従うだけなので、経験・引き継ぎは必要なく、人の代替が容易となる。また、全ての仕事はプログラムから管理され

るため、人の運用効率の数値化や進捗の管理などが可能となる。

プログラム化によって、作業をより適切な単位に分割し、逐次的に人に実行させられるようになる。複雑な作業をしている際、次に何をすれば良いのかといった情報がわからなくなることがあるが、これは、作業内容を覚えきれていなかったり、忘れてしまうといったことが原因で起こる。本提案のような仕組みを利用することによって、すべきことの記憶はコンピュータが担うことになる。人はその時すべきことをプログラムからの命令通りに動作するだけで良い。コンピュータが判断可能な部分に関しては全てコンピュータに委ねることができるため、人への負担を減らすことができる。

3.2 実世界プログラミング

人をセンサーやアクチュエータとして利用することで、現在一般的に使われているセンサーやアクチュエータでは実現困難な実世界プログラミングが可能となる。現在のセンサー技術では、その場の雰囲気を数値化・文字列化するなどのコンテキスト情報の分析は困難である。また、アクチュエータも単一の動きに特化したものが多く、複雑な動きを実現することは難しい。しかし、人をプログラム上でセンサーやアクチュエータとして利用できれば、既存のセンサーやアクチュエータでは難しい挙動も実現可能である。BabaScript 環境でならば、プログラム上で人はセンサーやアクチュエータと類似の挙動をすることができる。

また、人とセンサ・アクチュエータを状況に応じて使い分けるといったことも可能となる。コンテキスト情報を扱いそうなら人を利用し、温度などの数値を取得するだけならばセンサーを利用する、といった使い分けができる。センサー・アクチュエータがその場に存在するならばセンサー・アクチュエータを動作させるが、ない場合は人に命令する、といったことも実現可能である。

4. 関連研究

計算機では処理できないようなタスクを解決するために、人を計算資源としてプログラムに組み込む手法はヒューマンコンピュテーション [1] と呼ばれ、様々な研究が行われている。米 Amazon が運営している AmazonMechanicalTurk [2] は、クラウドソーシングのためのプラットフォームだ。mTurk API を通し、人間に対してタスクの実行を依頼することができる。AUTOMAN [3] は、crowdprogramming という概念を唱え、通常のプログラミング言語内でコンピュータによる計算と人による計算を統合した。CrowdForge [4] は、MapReduce のような機能をクラウドソーシングのためのフレームワークだ。クラウドソーシングするタスクを適切に分割し、人力で解かせた後、集合させるといったことができる。jabberwocky [5] は、クラウドソーシングプラットフォームを自由に作れる・再利用できる仕組みをもった Dormouse や MapReduce 的に人リソースを扱える ManReduce、SQL 風のスクリプト言語 Dog から構成される、クラウドソーシングのためのフレームワークだ。CrowdDB [6] では機械だけでは答えられないような DB へのクエリに対し、クラウドソーシングを使うことで返答させるための SQL ライクなプログラミングを提案している。CyLog [7] は Datalog に似たヒューマンコンピュテーションのためのプログラミング言語だ。人をデータソースとしてプログラムの中で利用する手法を提案している。これらの研究は、人を計算資源・データソースとして捉え、コンピュータの代替として人を利用している。本研究では、人の行動そのものをプログラムとして記述し、実行可能なものにすることを目的としている。

ユビキタスコンピューティングの研究分野においては、Human as Sensor といった概念も存在しており、研究が行われている。MoboQ [8] では、場所ベースの Q&A サービスを実装し、その効果を検証した。MoboQ ではプラットフォームとしてソーシャルメディアを利用しており、ソーシャルメディア上の人たちをセンサーとして利用している。スマートフォンを使ったセンシングのためのプラットフォームとしては、PRISM [9] などが発表されている。これらの研究では、人をセンサーとして利用し、情報を収集することを目的としている。本研究では、人の行動をプログラムとして記述することを目的としており、その利用方法はセンサーに限定されたものではない。

人のワークフローを定義する Web サービスとしては、atled [10] や Questetra [11] などが存在するが、これらのサービスは、人の行動をプログラムで記述するものではない。BabaScript 環境では、人・コンピュータの動作を同一のプログラム上で記述することが可能だ。

5. 考　　察

BabaScript 環境についての考察を行う。

5.1 命令の実行保障性

BabaScript 環境においては、命令の実行保障性が 100%保障されない。命令が表示されるインタフェースを見ていないことや、そもそも命令を無視するといった可能性が考えられ、そういったことが起きた際には命令実行が極端に遅くなったり、実行されないことがある。命令が来たことをわかりやすく通知したり、何かしらのインセンティブを、命令実行者に対して与える必要がある。

インセンティブとして考えられるものは、金銭を与えるか、実行者がメリットを享受するといったことだ。これは、どのようなフィールドで BabaScript 環境を運用するかによって、インセンティブの質が変わってくる。例えば、人プログラムの実行者と、クライアント側の実行者が労働関係にある場合、プログラムの実行がそのまま労働に繋がるため、BabaScript 環境のための特別なインセンティブが必要となるわけではない。家庭内などの利害関係があまりないような場面においてもインセンティブは大きな問題とはならない。

運用フィールド次第では、インセンティブが必要となるため、そういった場面においてはインセンティブを与える必要がある。BabaScript 環境では、現状では金銭などのインセンティブを与える仕組みは存在しないため、金銭などのインセンティブを与える必要がある場面においては運用が難しくなる。

5.2 命令の粒度

人への命令は様々な文言が想定される。具体的過ぎる文言の

命令であれば、プログラムは肥大化していき、命令に対して値を返す回数も増加することになり、クライアント側に大きな負担をかけることとなる。抽象的すぎる文言の命令であれば、人はどのように処理すれば良いのかわからなかったり、処理結果にブレが生じる危険性がある。

この文言が、どの程度の具体性を持ったものであると効果的なのかが重要となる。今後の運用を通して検討を進める。

5.3 複数の命令

同一の人への命令構文を含んだプログラムが複数実行された場合、プログラムごとに異なるコンテキストの命令が交じる可能性がある。例えば、料理と洗濯という二つのプログラムが実行された場合、料理をしている最中に「洗濯物をしまい込め」といった命令が配信される、といったことが起こりうる。

一人の人リソースは一つのプログラムだけが利用できるようにすることで、この問題は解決可能である。baba という人リソースをプログラム A が利用中の場合は、プログラム B から baba という人リソースを使えないようにすれば、コンテキストの混じった命令が行われることはない。今後の課題として検討をする。

6. おわりに

本論文では、人の行動を記述可能なプログラミング環境 BabaScript を提案した。人力処理構文と命令を受け取り値を返すことのできるクライアントアプリケーションを組み合わせることによって、プログラム上で人を表現することが可能になった。BabaScript 環境においては、人とコンピュータの双方は同じプログラム内で動きを定義することができる。また、BabaScript 環境の応用例について述べ、利用時に起こりうる問題などについて考察を行った。今後は、課題の解決と応用例の実装、有用性の検証を行う。

7. 謝　辞

本研究は、独立行政法人情報処理推進機構の「2013 年度未踏 IT 人材発掘・育成事業」の支援を受けて開発を行っている。

文　献

[1] L. von Ahn. 2007. Human computation. In Proceedings of the 4th international conference on Knowledge capture. K-CAP '07. ACM.
[2] Amazon Machanical Turk http://www.mturk.com
[3] Barowy, D. W., Curtsinger, C., Berger, E. D., andMcGregor, A. AutoMan: A Platform for Integrating Human-Based and DigitalComputation. In Proc. OOPSLA (2012).
[4] A. Kittur, B. Smus, and R. E. Kraut. CrowdForge: Crowdsourc- ing Complex Work. Tech. Rep. CMU-HCII-11-100, Human- Computer Interaction Institute, School of Computer Science, Carnegie Mellon University, February 2011.
[5] S. Ahmad, A. Battle, Z. Malkani, and S. Kamvar. The Jabberwocky Programming Environment for Structured Social Computing. In UIST, pp. 5364, 2011.
[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: Answering Queries with Crowdsourcing. In SIGMOD, pp. 6172, 2011.
[7] Atsuyuki Morishima, Norihide Shinagawa, Tomomi Mitsuishi, Hideto Aoki, Shun Fukusumi. " CyLog/Crowd4U: A Declarative Platform for Complex Data-centric"
[8] Liu, Yefeng and Alexandrova, Todorka and Nakajima, Tatsuo. Using Stranger As Sensors: Temporal and Geo-sensitive Question Answering via Social Media. Proceedings of the 22Nd International Conference on World Wide Web, pp. 803–814, 2013
[9] Das, Tathagata and Mohan, Prashanth and Padmanabhan, Venkata N. and Ramjee, Ramachandran and Sharma, Asankhaya. PRISM: Platform for Remote Sensing Using Smartphones. Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. pp.63–76, 2010.
[10] https://www.atled.jp/
[11] http://www.questetra.com/
[12] https://github.com/node-linda/node-linda-base